



UNIVERSIDAD DEL VALLE
Escuela de Ingeniería de Sistemas y Computación

Desarrollo de un Modelo HTM para la Ejecución de Tareas Utilizando el Lenguaje Natural

Trabajo de Grado para optar por el título de Ingeniero de Sistemas y
Computación

Presentado por:
Sebastián Narváez Rodríguez
sebastian.narvaez@correounivalle.edu.co
1035605

Dirigido por:
Ángel de la E. García & Raúl E. Gutierrez

Junio de 2016

¿Agradecimientos?

Siempre he tenido la impresión de que esta es la sección que nadie lee, a menos que se trate de un evaluador del trabajo, o aspire a que su nombre aparezca aquí. Lo cierto es que, aunque es éticamente correcto agradecer a quienes nos han ayudado, la mención de un nombre en esta sección no provee al nombrado de ningún reconocimiento o beneficio más allá, quizá, de una moderada satisfacción. Por lo tanto, esto se ha convertido más que todo en una cuestión de protocolo.



Figura 1: Los protocolos son para el internet. Imagen obtenida en Pixbay.

Entre los primeros capítulos de «El Gen Egoísta», de Richard Dawkins, el autor se cuestiona acerca del altruismo, la ética y las reglas de convivencia en una sociedad. Uno de los planteamientos va más o menos así: Muchas personas trazan una línea entre los humanos y los demás seres vivos, teniendo los primeros ciertos beneficios presupuestos sobre los otros. Algunos justifican esta distinción como un resultado de la simpatía por los semejantes. Sin embargo, dos personas pueden llegar a ser muy diferentes. Además, bajo el principio de la simpatía, ¿por qué no pueden considerarse también a otros homínidos, como gorilas y orangutanes, merecedores de tales tratos especiales? Y extendiendo la misma idea, ¿por qué no todos los mamíferos? En fin, ¿cómo se define el punto donde dos seres comienzan a ser «demasiado diferentes»?

El fragmento del que he comentado se me vino a la cabeza porque me encuentro en

un dilema parecido. Todos los días, cada persona¹ crea una inmensa red de relaciones que le llevan, sea placentera o aparatosamente, a cierto resultado. El hecho de que esté escribiendo esto significa que cada situación que he experimentado hasta ahora me ha permitido, e incluso incentivado, a escribirlo. Así que, ¿cómo trazo la línea que me permita escoger las situaciones o personas a las que incluir en esta sección?

Supongo que podría empezar por Angel García, quien fue el que recomendó «El Gen Egoísta» en una de sus clases (permitiendo la referencia anterior), y quien me dio a conocer las investigaciones de Jeff Hawkins, una de las bases de esta obra. Además, él y Raúl E. Gutierrez fueron los directores de este trabajo de grado. Ambos estuvieron pendientes de los progresos y aportaron valiosas ideas. Raúl también fue quien me introdujo al mundo del procesamiento del lenguaje natural, transmitiéndome un poco de su pasión en el proceso. Con sus trabajos aprendí el lenguaje de programación Python, que me cayó como anillo al dedo para la implementación de HTM-TEUL.

A pesar de que no los conocí en persona, el equipo de Numenta fue un gran apoyo para entender la teoría HTM y el funcionamiento de Nupic. Su tiempo de respuesta y amabilidad siempre me sorprendieron. Gracias a ellos pude tener una experiencia más cercana con un proyecto de código abierto, lo cual fue muy enriquecedor.



Figura 2: La espiral de la procrastinación. Imagen obtenida en Pixbay.

Familia, amigos ²... Las relaciones con las personas cercanas son complejas. «Gracias por todo su apoyo», sí, pero así como hay momentos de placer también los hay de desesperación. Todos ellos me dieron momentos de tranquilidad que propiciaron la concentración y un fluido desarrollo del trabajo, y momentos de caos que me llevaron hacia la espiral de la procrastinación. También hubo jornadas de debate, que aunque

¹De hecho, esto aplica para cualquier ente, pero me concentraré arbitrariamente en las personas para facilitar las cosas

²En especial, agradezco a Paula A. Palacios, quien se ofreció repetidas veces a leer este documento y me dio múltiples sugerencias (por ejemplo, animar esta sección con imágenes).

no aportaran directamente ideas para este trabajo, ampliaron mis perspectivas en temas variados. Tuve momentos cercanos a la muerte, y otros tantos donde me encontré rodeado de muchas clases distintas de vida. Pensándolo bien, frecuentemente fue en estos tiempos de dispersión en que descubrí las soluciones que me ayudaron a salir de los bien conocidos estancamientos mentales. Así que gracias, procrastinación.

Gracias a las voces guturales del Univalle Metal Fest, que acaban de despertarme para terminar los agradecimientos. Llevo aplazándolos desde hace meses. El bajo alcanza a retumbar hasta evaluar, y la enérgica batería acelera el ritmo de mi escritura.

Índice general

Agradecimientos	1
Resumen	6
1 Introducción	7
2 Objetivos	9
2.1 Objetivo General	9
2.2 Objetivos Específicos	9
3 Marco Teórico	10
3.1 Procesamiento de Lenguaje Natural (PLN)	10
3.2 Red Neuronal Artificial (RNA)	10
3.3 Hierarchical Temporal Memory (HTM)	11
3.4 Algoritmos Evolutivos	13
4 Estado del Arte	14
4.1 Sistemas de Ejecución de Tareas	14
4.2 Representación Semántica del Lenguaje	14
4.3 PLN y la Teoría HTM	15

5	Trabajo Realizado	16
5.1	La Interfaz Gráfica	16
5.2	El Conjunto de Entrenamiento	18
5.3	Los Codificadores	19
5.3.1	Custom Category Encoder (CCE)	19
5.3.2	Randomized Letter Encoder (RLE)	19
5.3.3	Totally Random Encoder (TRE)	20
5.4	El Clasificador	21
5.5	Los Modelos HTM	21
5.5.1	Modelo Jerárquico Clásico	22
5.5.2	Modelo Uni-Nivel	23
5.5.3	Modelo Feedback	23
5.6	Evaluación de los Modelos	25
5.7	Trabajo Adicional	25
5.7.1	Pyrams Finder	25
6	Resultados y Discusión	27
7	Conclusiones	34
8	Trabajos Futuros	35
9	Epílogo	36

Resumen

¿Cansada de que sus máquinas no le entiendan? ¿Desearía poder enseñarles a ejecutar nuevas órdenes, de nuevas formas, en lugar de quedarse con el conjunto pre-cargado? Este proyecto podría ser la solución. Utilizando la revolucionaria teoría HTM de Jeff Hawkins y los Cortical Learning Algorithms de Numenta, se ha desarrollado un software capaz de asociar oraciones en español (¡Y hasta en inglés!) con «eventos» de un mundo virtual. Además, como se dijo desde el principio, podrá enseñarle nuevas formas de referirse a un evento. No lo piense más, ¡Ensaye HTM-TEUL y denos su opinión! Este es apenas una demostración de lo que podría hacerse utilizando estas nuevas herramientas de la inteligencia artificial. Nuestros investigadores seguirán experimentando y trabajando para superar el record de 89% de aciertos obtenido en las pruebas. Así que corra a su navegador más cercano y busque el proyecto en github. Así es, ¡es de código abierto!

– Muy bien. Pero olvidaste mencionar los diferentes modelos que fueron probados, con no tan favorables porcentajes de aciertos. ¿Y qué hay de los codificadores? ¡Son indispensables para la teoría HTM! Y no se pudo encontrar uno que reflejara bien las características semánticas de las palabras o eventos; los mejores resultados siguieron apareciendo cuando se trataba cada uno por separado, sin ninguna relación de significado. Además, ¿Cómo que «investigadores»? Sólo tenemos a un «ingeniero», y de a medio tiempo...

– Calma, calma. Esta sección es para vender el producto. Si a alguien de verdad le interesan esos detalles, ya leerá ese aburrido documento técnico.

– De aburrido nada, ¡ahí está toda la esencia! Por favor, al menos aclara que irónicamente, aunque se puedan añadir nuevas formas de dar las mismas órdenes, no se puede añadir nuevas órdenes, al menos en esta versión. Así como está suena a publicidad engañosa.

– Ya, le pondré un asterisco con letra pequeña.

– ... Mejor nos callamos y dejamos que la gente lea el documento, ¿sí?

Capítulo 1

Introducción

Imagínese, por un momento, recostado en una silla supremamente cómoda. De repente, siente unas ganas inmensas de tomarse un jugo frío, pero la nevera está muy lejos y levantarse ahora, luego de una larga jornada, le parece casi un crimen. Entonces se aclara la garganta y, con voz firme, dice «*Computadora, un jugo de naranja con hielo, por favor*». En ese momento escucha unos sonidos metálicos y, al cabo de unos segundos, una mano robótica le alcanza justo lo que pidió. ¿Se le hace conocida la situación? Seguramente alguna vez se habrá imaginado algo parecido, o al menos lo habrá visto o leído en una historia de ciencia ficción.

El que una máquina pueda entender las palabras humanas y actuar conforme a ellas ha sido, desde hace mucho, un deseo y un constante objeto de estudio en las Ciencias de la Computación. El área que se encarga de este tema es llamada Procesamiento de Lenguaje Natural (PLN). Tradicionalmente, en el PLN se han utilizado modelos probabilísticos que se basan en corpus anotados con características sintácticas del lenguaje [9]. Tales modelos permiten operaciones útiles como la desambiguación del sentido de las palabras en un contexto determinado (Word Sense Desambiguation), pero su flexibilidad y resistencia ante el ruido puede verse comprometida, ya que dependen del corpus con el cual fueron entrenados. La fabricación de dichos corpus, por otra parte, requiere de expertos en el lenguaje en cuestión que anoten manualmente una gran cantidad de sentencias, proceso que puede ser costoso y lento.

Jeff Hawkins se dió cuenta de que, no solo en el PLN sino en todo el campo de la Inteligencia Artificial (IA), suelen utilizarse algoritmos que son muy buenos para una función específica, pero tienen dificultades al enfrentarse a problemas más generales. Tal aproximación tiene una enorme utilidad comercial, pero deja a un lado una propuesta de investigación interesante: Crear máquinas «verdaderamente inteligentes» (es decir, que sean capaces de aprender y razonar de forma similar a como lo hace un animal) [8]. Con esta idea en mente, Hawkins y sus socios fundaron Numenta, una compañía que intenta realizar ingeniería inversa al neocórtex humano. Este esfuerzo dio nacimiento a la teoría de la Memoria Jerárquica Temporal (Hierarchical Temporal Memory ó HTM),

que incluye algoritmos de aprendizaje multipropósito (Cortical Learning Algorithms ó CLA) capaces de encontrar patrones espacio-temporales en, y entre la información obtenida desde diferentes sensores [7].

En este trabajo se implementó un Sistema de Ejecución de Tareas (llamado HTM-Teul ¹) utilizando Nupic, la implementación oficial y de código abierto de los CLA [17]. Para probar dicho sistema, se implementó también un mundo virtual con un conjunto determinado de eventos posibles. HTM-Teul recibe una oración en español, y en base a ella trata de predecir y ejecutar un evento asociado en el mundo virtual.

Si bien es cierto que el entrenamiento de este sistema también puede llegar a ser costoso y lento, el concepto de ‘entrenamiento’ aquí es flexible. Esto es, las etapas de entrenamiento y uso del sistema no están completamente separadas, sino que son procesos paralelos y que pueden aportarse el uno al otro. Además, como es el usuario quien provee la información de entrenamiento, su experiencia será más acorde a sus necesidades y costumbres.

¹Puede ver el código fuente en <https://github.com/larvasapiens/htm-teul>

Capítulo 2

Objetivos

2.1 Objetivo General

Implementar y probar la validez (medida en porcentaje de aciertos) de un sistema de ejecución de tareas, que utiliza la herramienta NuPic para formar relaciones entre oraciones escritas en español, y eventos de un mundo virtual creado para este trabajo.

2.2 Objetivos Específicos

1. Construir un mundo virtual en el que se puedan llevar a cabo un conjunto de acciones finitas, bien diferenciadas y cuya ejecución pueda ser advertida por un usuario humano sin limitaciones sensoriales.
2. Adaptar las cadenas de texto escritas en español y los eventos producidos en el mundo virtual a una representación válida para su procesamiento con los algoritmos de NuPic (Codificación).
3. Analizar la validez (medida en porcentaje de aciertos) del sistema de ejecución de tareas.

Capítulo 3

Marco Teórico

3.1 Procesamiento de Lenguaje Natural (PLN)

El Procesamiento de Lenguaje Natural es un área de estudio que combina conocimientos de diversos campos como las ciencias de la computación, la lingüística y la probabilidad y estadística. Se dedica al estudio de la interacción humano-computadora, enfocándose en la construcción de sistemas que toman el lenguaje natural como elemento de entrada y/o salida [4]. Entre las aplicaciones más destacadas del PLN se encuentran traductores (Ej.: Google Translate), sistemas de pregunta-respuesta (Ej.: Wolfram Alpha) y sistemas de ejecución de tareas (Ej.: Comandos de Voz en Android).

3.2 Red Neuronal Artificial (RNA)

Una Red Neuronal Artificial es un sistema de aprendizaje que está compuesto por un conjunto de unidades (llamadas Neuronas) conectadas entre sí de una forma determinada [18]. Se utilizan para encontrar patrones en un set de entrenamiento (datos de entrada con sus respectivas salidas), de forma que luego puedan predecir una salida a partir de una un dato o conjunto de datos. Las Redes Neuronales Artificiales son consideradas como la rama de la IA cuya aproximación imita la biología. Esto les provee flexibilidad y generalidad en comparación a las aproximaciones más tradicionales, pero a cambio suelen ser más lentas e inexactas [8].

3.3 Hierarchical Temporal Memory (HTM)

El cerebro humano (más concretamente el neocórtex) está diseñado para identificar y aprender patrones espacio-temporales a partir de un flujo de datos. Tomando como ejemplo la visión, el ojo recibe una secuencia de imágenes que varían con el paso del tiempo. El cerebro es capaz de realizar abstracciones y diferenciar patrones tanto a nivel espacial (bordes, colores, y a un más alto nivel, objetos o personas) como temporal (una persona caminando, un ave volando) [8]. Una HTM posee estructuras y algoritmos que replican tal funcionamiento, y por ello puede ser catalogada como una Red Neuronal Artificial. Sin embargo, la aproximación tradicional de éstas últimas es diferente.

El proceso de aprendizaje de una HTM incluye la formación de Representaciones Dispersas Distribuidas (Sparse Distributed Representations / SDRs) para cada elemento del flujo de datos que la HTM tiene como entrada. Las SDRs deben su nombre a que distribuyen la información y utilizan pocos bits para representar un dato. Esto permite que un error local no tenga gran impacto en el significado global de la representación, y que se requiera sólo un subconjunto de sus bits para identificar, con buena certeza, el dato original [7].

La unidad principal de memoria y predicción de una HTM es llamada región (Ilustrada en la figura 3.1). Una región se compone de columnas, y cada columna a su vez se compone de células (análogas a las neuronas del cerebro). Una HTM puede tener muchas regiones organizadas en una jerarquía. Las regiones más bajas reciben directamente los datos, mientras que las más altas procesan información con un más alto nivel de abstracción. Esto permite reutilizar la información básica ya aprendida para conceptos diferentes. Volviendo a la analogía de la visión, el cerebro sólo necesita aprender una vez qué es un borde, y luego es capaz de aplicar el mismo concepto para cualquier objeto que vea [8].

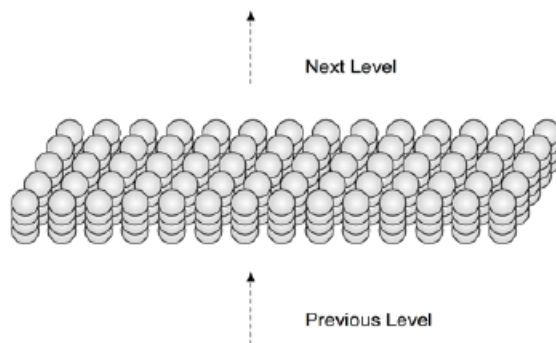


Figura 3.1: Parte de una región de HTM. Extraída de [7]

Las regiones se construyen mediante dos procesos: El Agrupamiento Espacial (Spatial Pooling) y el Agrupamiento Temporal (Temporal Pooling).

Agrupamiento Espacial

El Agrupamiento Espacial es llevado a cabo por el Spatial Pooler (SP). Cuando una región recibe un dato, el Spatial Pooler forma conexiones entre cada bit activo de la entrada y algunas columnas de la región. Estas conexiones son llamadas Sinapsis, y cada una de ellas tiene un valor de permanencia que determina si es o no una Sinapsis válida. Las columnas que han formado una Sinapsis válida con un bit activo son a su vez activadas, inhibiendo la activación de sus columnas vecinas dentro de un radio de inhibición. El conjunto de columnas activas al final de este procedimiento es una representación ‘dispersa’ del dato de entrada. Las entradas parecidas tendrán representaciones parecidas [7].

Agrupamiento Temporal

En el Agrupamiento Temporal, llevado a cabo por la Temporal Memory (TM), se forman representaciones de los datos de acuerdo al contexto (que se da por las entradas anteriores). Mientras que el SP opera a nivel de columnas, el TM lo hace a nivel de células. Cuando una célula es activada, esta forma conexiones con las células vecinas que estuvieron activas debido a la entrada anterior. Tales conexiones son luego utilizadas para predecir qué entrada ocurrirá a continuación [6]. Un ejemplo de la representación de una entrada en contexto de las entradas anteriores se muestra en la figura 3.2. Las células en gris claro fueron activadas por la entrada actual, mientras que las células en gris oscuro se encuentran en estado predictivo.

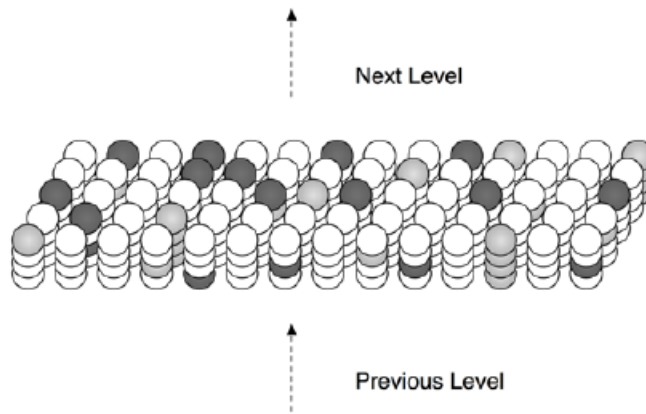


Figura 3.2: Representación de una entrada en contexto de las entradas anteriores. Extraída de [7]

Como cada columna tiene varias células, y cada entrada es representada por varias columnas, es posible representar una gran cantidad de contextos por entrada de forma única. Además, es posible tener en cuenta un historial de muchas entradas anteriores, en lugar de sólo la entrada inmediatamente anterior [6].

3.4 Algoritmos Evolutivos

Son algoritmos de optimización cuyo proceso de búsqueda de soluciones se basa en la teoría de la evolución. Se empieza con una población aleatorizada de Individuos (posibles soluciones), que pasan por un proceso de Selección, en donde la probabilidad de ser elegidos está relacionada con la calidad de la solución que representan. A los Individuos seleccionados se les aplican mutaciones (generalmente cambios pequeños), cruces, u otras operaciones que introduzcan variabilidad, para así conformar una nueva población. El proceso entonces vuelve a iniciar con la nueva población. Este tipo de algoritmos tiene una parte de aleatorio y otra de determinista, por lo que sacrifican un poco de eficiencia a favor de una mayor robustez, pero manteniendo cierto control sobre la dirección hacia donde se dirige la búsqueda de soluciones [5].

Capítulo 4

Estado del Arte

4.1 Sistemas de Ejecución de Tareas

Quizá los ejemplos de sistemas de ejecución de tareas más famosos son Siri, para iOS, y Google Voice Search, para Android. Sin embargo, el fuerte de estas aplicaciones es su reconocimiento de voz. Existe además un software open-source llamado Betty¹, que recibe una oración en lenguaje natural (sólo en inglés hasta la versión actual - 0.1.8) y ejecuta el comando correspondiente en la consola. La diferencia fundamental entre las aplicaciones anteriormente nombradas y HTM-Teul es que éste último permite añadir oraciones equivalentes o sinónimas que ejecuten las tareas existentes.

4.2 Representación Semántica del Lenguaje

La idea de representar las palabras por sus características semánticas no es nueva. WordNet es una famosa base de datos en inglés (aunque existen versiones alternativas en otros idiomas) que almacena relaciones semánticas entre palabras, como de sinonimia o hponimia. La cercanía entre dos palabras en la red resultante depende de qué tan fuerte es su relación en un contexto determinado [12]. WordNet ha sido utilizado en diversas aplicaciones, de las cuales se destaca la desambiguación conceptual en tareas de extracción de información [13].

Además de aplicaciones finales, se han construido modelos para la generación de relaciones semánticas utilizando bases de datos como WordNet. En [1] se propone el uso de la lógica probabilística para cumplir tareas de Textual Entitlement y Textual Similarity. Este tipo de aproximaciones permite propiedades interesantes como la gene-

¹Puede ver el proyecto en <https://github.com/pickhardt/betty> (Revisado el 10 de Febrero de 2016)

ración dinámica de ontologías que contienen únicamente los elementos requeridos para formar las relaciones (y sus probabilidades) entre las palabras o frases en cuestión. Las ontologías pueden ser entonces utilizadas por un mecanismo de inferencia que cumple una tarea específica [10, 2].

Las representaciones vectoriales distribuidas que representan características semánticas de las palabras han sido uno de los objetos de estudio cuya popularidad ha ascendido junto con el auge de las técnicas deep learning. Ejemplos de tales trabajos son [3, 11]. Con esta aproximación los vectores correspondientes a palabras similares tienen una distribución de valores similar.

4.3 PLN y la Teoría HTM

En la wiki de Nupic se declara que su implementación actual no es ideal para tareas complejas de PLN². Sin embargo, la Start-up Estadounidense Cortical.io construyó un marco teórico para el PLN llamado Semantic Folding Theory, que se basa en algunos fundamentos de la Teoría [19]. La implementación de esta teoría está a disposición del público bajo la Retina API, y produce SDRs (Sparse Distributed Representations) compatibles con Redes HTM como las que se pueden construir con Nupic. Los SDRs que produce la herramienta de Cortical, llamados Semantic Fingerprints, son matrices binarias donde los bits cercanos representan conceptos parecidos o relacionados [19].

²Puede ver la sección referida aquí <https://github.com/numenta/nupic/wiki/Natural-Language-Processing>. Revisado el 16 de Febrero de 2016

Capítulo 5

Trabajo Realizado

La primera parte de este documento refleja una gran porción del trabajo realizado: la lectura de múltiples artículos y whitepapers. Ahora que estamos en contexto, nos concentraremos en lo que se refiere a la implementación del software HTM-TEUL, las conexiones que tiene con la teoría HTM y las ideas que soportan sus mecanismos subyacentes.

Antes de continuar, vale la pena conocer algunos detalles técnicos. HTM-TEUL está implementado completamente en Python 2.7. Utiliza la librería gráfica del framework QT5 ¹, a través de los bindings PyQt5 ², y la implementación oficial de los CLA, Nupic ³. Para las librerías externas mencionadas, se utilizó su opción de licencia de uso libre.

5.1 La Interfaz Gráfica

En lugar de ahondar inmediatamente en el sistema de aprendizaje, es conveniente revisar primero la interfaz gráfica. De esta forma, tendremos una perspectiva más tangible del software implementado.

Al abrir HTM-TEUL, nos encontramos con la ventana ilustrada en la figura 5.1. A la izquierda se encuentra el Mundo Virtual, una grilla de 5x5 con casillas llenas de pasto. Nuestra protagonista es una carismática hormiga ⁴.

Los Controles del personaje, ubicados en la parte superior derecha, permiten visualizar los que en este trabajo son conocidos como «eventos del mundo virtual». Los botones «Arr.», «Aba.», «Izq.» y «Der.», mueven a la hormiga en las direcciones Arriba,

¹<http://www.qt.io/download/>

²<https://www.riverbankcomputing.com/software/pyqt/download5>

³<https://github.com/numenta/nupic> - URLs visitadas el 8 de Junio de 2016

⁴Especie: *Ectatomma ruidum*

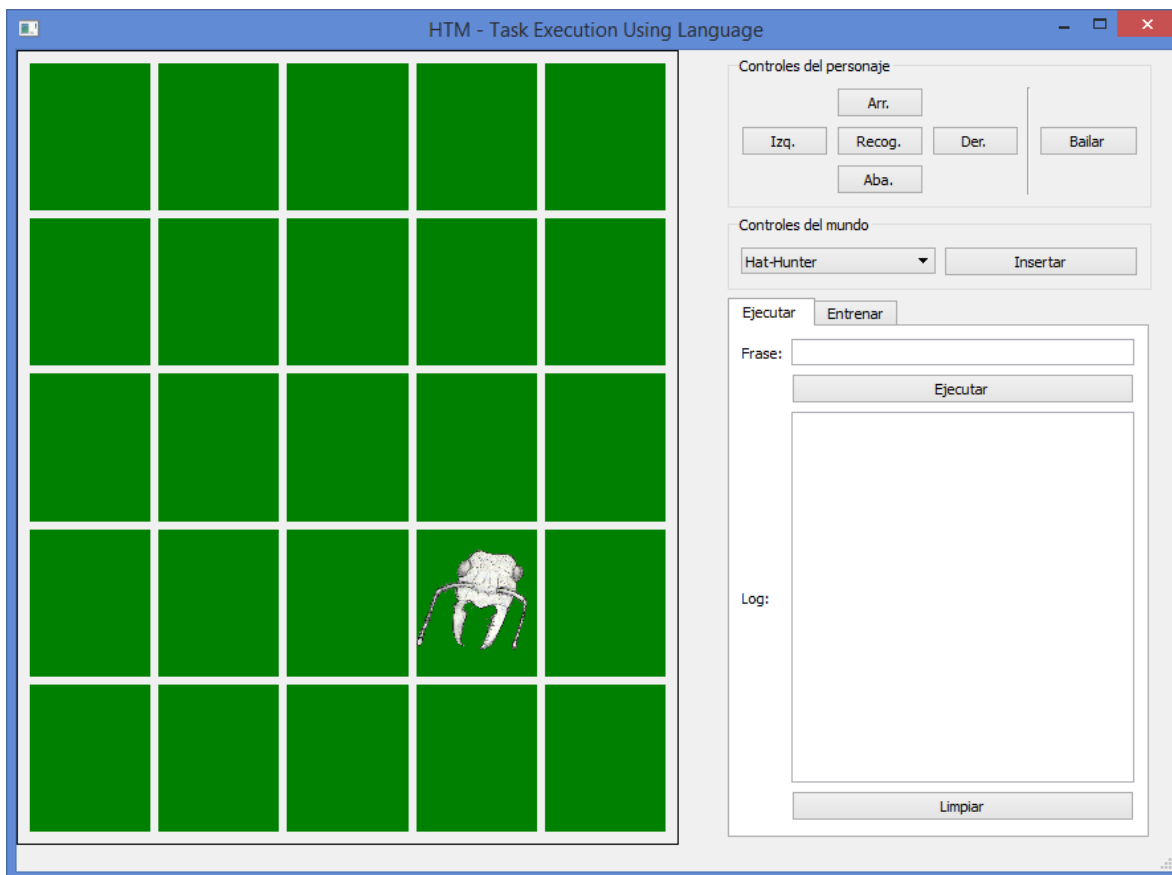


Figura 5.1: Interfaz Gráfica de HTM-TEUL

Abajo, Izquierda y Derecha respectivamente. Con el botón «Recog.», la hormiga recoge el objeto que esté en su misma casilla, y con el botón «Bailar», la hormiga baila por unos segundos.

Los Controles del mundo, que se encuentran debajo de los Controles del personaje, cuentan con una única opción: ubicar un objeto en una posición aleatoria del mundo virtual. Es necesario que haya un objeto en alguna casilla para poder visualizar el evento de «Recoger».

Finalmente, la parte inferior derecha se encuentra dividida en dos pestañas: «Ejecutar» y «Entrenar». En la primera hay un espacio para introducir una Frase u Oración, un botón para ejecutarla y un log que lleva un registro de lo que ha pasado en la sesión. Cuando se oprime el botón «Ejecutar», la oración escrita es pasada al Modelo HTM, el cual predice un evento del Mundo Virtual, que es a su vez visualizado en la Interfaz Gráfica.

La segunda pestaña, «Entrenar», es ilustrada en la figura 5.2. Aquí se puede entrenar el Modelo HTM con nuevas oraciones en tiempo de ejecución. Esta es una característica clave, ya que permite al usuario adaptarlo a su gusto.

The interface is titled 'Entrenar' (Train). It contains the following elements:

- Input Fields:**
 - Frase:** agite la melena
 - Accion:** Bailar (dropdown menu)
- Buttons:**
 - Añadir:** Add button
 - Limpiar Tabla:** Clear Table button
 - Entrenar:** Train button
- Table:**

	Frase	Accion	Argumento
1	muevase pa arriba	mover-event	arriba-event
2	agite la melena	bailar-event	nothing-event
- Iterations:** Iteraciones 10

Figura 5.2: Pestaña «Entrenar» de HTM-TEUL

5.2 El Conjunto de Entrenamiento

Todo sistema de aprendizaje requiere ser entrenado. Para ello, usualmente se dispone de una colección de datos estructurados, los cuales guardan ciertos patrones. Estos patrones son, esencialmente, lo que el sistema aprende. A fin de construir este conjunto de entrenamiento, se pidió a varias personas que anotaran diferentes formas de describir los eventos del mundo virtual.

La recolección de datos se llevó a cabo utilizando la plataforma Google Drive, que permite a varias personas editar un documento de forma simultánea. En la tabla 5.1 se muestran varios ejemplos de los datos recogidos. A pesar de que el objetivo era utilizar oraciones escritas en español, se obtuvieron también algunas en inglés.

Oración	Seq. de Eventos
a mover el esqueleto	bailar-event nothing-event
shake your booty	bailar-event nothing-event
muevete hacia el norte	mover-event arriba-event
desplazate para la izquierda	mover-event izquierda-event
por favor levanta el sombrero	recoger-event nothing-event

Cuadro 5.1: Subconjunto de los datos recogidos para el entrenamiento

5.3 Los Codificadores

Uno de los objetivos de este trabajo es convertir las oraciones y los eventos del mundo virtual en entradas aptas para los modelos HTM. Estos modelos reciben patrones de activación de bits, que en Nupic son representados como arrays binarios.

En este trabajo, tanto los eventos como las palabras son tratados como cadenas de texto. A continuación se presentan los Codificadores utilizados, que se encargan de convertir cadenas de texto a arrays binarios.

5.3.1 Custom Category Encoder (CCE)

El Category Encoder está incluido en Nupic por defecto [17] y su funcionamiento es sencillo. A cada elemento de una lista de categorías (cadenas de texto) se le asignan ‘w’ bits activos. Los bits activos de una categoría nunca se superponen a los de otra categoría. Esto significa que, desde un punto de vista semántico, cada categoría es tratada como si no tuviese nada en común con las demás. Tal resultado es útil para este trabajo porque no asume ninguna relación indeseada entre dos entradas, pero al mismo tiempo puede hacer que el aprendizaje sea más lento.

Se realizó la implementación del Custom Category Encoder para HTM-TEUL. Este codificador extiende la funcionalidad del Category Encoder, permitiendo que se reserve espacio para n categorías adicionales que no sean conocidas al momento de su creación. Este comportamiento es deseable ya que diferentes personas utilizan diferentes palabras para referirse a las mismas cosas, y el sistema debería ser flexible ante esta característica de los lenguajes.

5.3.2 Randomized Letter Encoder (RLE)

Este codificador fue desarrollado específicamente para este trabajo. Primero, cada letra de la palabra es pasada a binario utilizando un Category Encoder cuya lista de categorías son las letras en minúscula del alfabeto inglés (por esta razón, no soporta

```

encoder = CustomCategoryEncoder(w=3, categoryList=["a", "b"],
                                nAdditionalCategorySlots=1, forced=True)

encoder.encode("a")
#Out (bits asignados al momento de creación):
array([0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0], dtype=uint8)

encoder.encode("z")
#Out ("z" no estaba en la lista de categorías, se le asigna un
# espacio reservado):
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1], dtype=uint8)

encoder.encode("c")
#Out (categoría <<desconocida>>, no hay más espacios reservados):
array([1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=uint8)

```

Listing 1: Ejemplo de uso del Custom Category Encoder

acentos). Al final del array binario resultante, se añaden en posiciones aleatorias un número especificado de bits activos. Con este Codificador se intenta dar cierta cercanía a las representaciones de palabras que se escriben parecido, pero conservando unos bits aleatorios para minimizar los fallos por palabras homónimas.

```

encoder = RandomizedLetterEncoder(width=60, nRandBits=2, actBitsPerLetter=1)

encoder.encode('le')
# Out (Bits activos: 12 ('l'), 32 ('e'), 54 y 57 (Aleatorios)):
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
↪ 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
↪ 0, 0, 0, 1, 0, 0, 1, 0, 0], dtype=uint8)

encoder.encode('la')
# Out (Bits activos: 12 ('l'), 28 ('a'), 55 y 58 (Aleatorios)):
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
↪ 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
↪ 0, 0, 0, 0, 1, 0, 0, 1, 0], dtype=uint8)

```

Listing 2: Ejemplo de uso del Randomized Letter Encoder

5.3.3 Totally Random Encoder (TRE)

El Totally Random Encoder genera posiciones aleatorias para el número n de bits activos que se escoja. La primera vez que recibe una entrada, el codificador guarda las posiciones generadas. De esta forma, devolverá la misma codificación las próximas veces que reciba dicha entrada. Si el tamaño del array escogido es grande y el n pequeño, se vuelve poco probable que dos entradas compartan bits activos en su codificación. Como

resultado se obtendrá un efecto parecido al del Category Encoder, pero los bits estarán más dispersos, una propiedad deseable según la teoría HTM.

```
encoder = TotallyRandomEncoder(width=15, nActiveBits=3)

encoder.encode("a")
#Out (bits asignados al momento de creación):
array([0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0], dtype=uint8)

encoder.encode("r")
#Out (bits asignados al momento de creación):
array([0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1], dtype=uint8)
```

Listing 3: Ejemplo de uso del Totally Random Encoder

5.4 El Clasificador

Los algoritmos de Nupic son alimentados con versiones codificadas de los datos [7]. Por lo tanto, se necesita «decodificar» sus salidas para obtener predicciones que puedan ser leídas por un humano. Este proceso no es trivial. De hecho, Nupic proporciona una herramienta llamada CLAClassifier para mapear la salida de la Temporal Memory (que es una lista de las células activas), con su respectivo valor real [14]. Este Clasificador no tiene una base biológica sino que, al igual que las técnicas de machine learning tradicionales, intenta dar la predicción más probable.

Para este trabajo, se modificó el CLA Classifier de Nupic para que tome en cuenta únicamente los valores reales pertenecientes a los eventos del mundo virtual. De esta forma, el Clasificador de HTM-TEUL siempre predice eventos, en vez de una combinación de eventos y palabras.

5.5 Los Modelos HTM

Durante el desarrollo de este trabajo se trató de pensar en cómo un cerebro podría aprender el Lenguaje Natural. De estas meditaciones surgieron varias hipótesis ⁵, que sirvieron como base para los diferentes modelos que se probaron. Tales modelos reciben sus entradas desde dos sensores: Uno correspondiente a las palabras de una oración, y el otro a una secuencia de eventos en el Mundo Virtual.

A continuación se explica cada una de dichas hipótesis y se muestran los resultados

⁵Cabe aclarar que estas hipótesis no tienen base biológica, pedagógica o académica de ningún tipo. Son únicamente el resultado de «sesiones filosóficas» del autor

que tuvieron los modelos con las diferentes pruebas aplicadas ⁶. A los modelos les fueron dados nombres que tratan de describir su estructura, pero no tienen un significado intrínseco.

5.5.1 Modelo Jerárquico Clásico

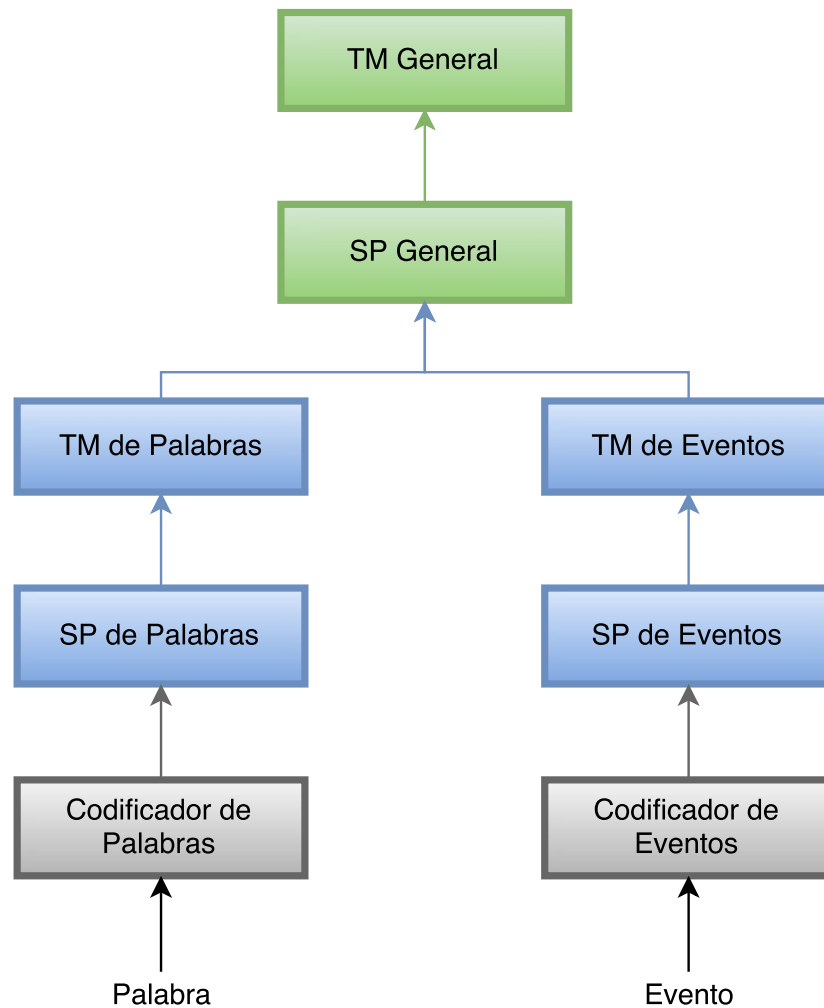


Figura 5.3: Estructura del Modelo Jerárquico Clásico

Según la teoría de HTM, las entradas que vienen desde los sensores serían procesadas por el cerebro independientemente a un bajo nivel, para luego inter-relacionarlas con cierto nivel de abstracción.

La figura 5.3 ilustra una estructura HTM que corresponde a este razonamiento, donde el nivel más bajo está representado con azul, y el más alto con verde. La “TM

⁶Una comparación de estos resultados se encuentra en el capítulo 7

de Palabras” aprende el orden en que deben de ir las palabras en una oración para que ésta tenga sentido, mientras que la “TM de Eventos” aprende las posibles secuencias de eventos del mundo virtual. “SP General” y “TM General” unen esas dos abstracciones y tratan de encontrar la relación que existe entre ellas.

5.5.2 Modelo Uni-Nivel

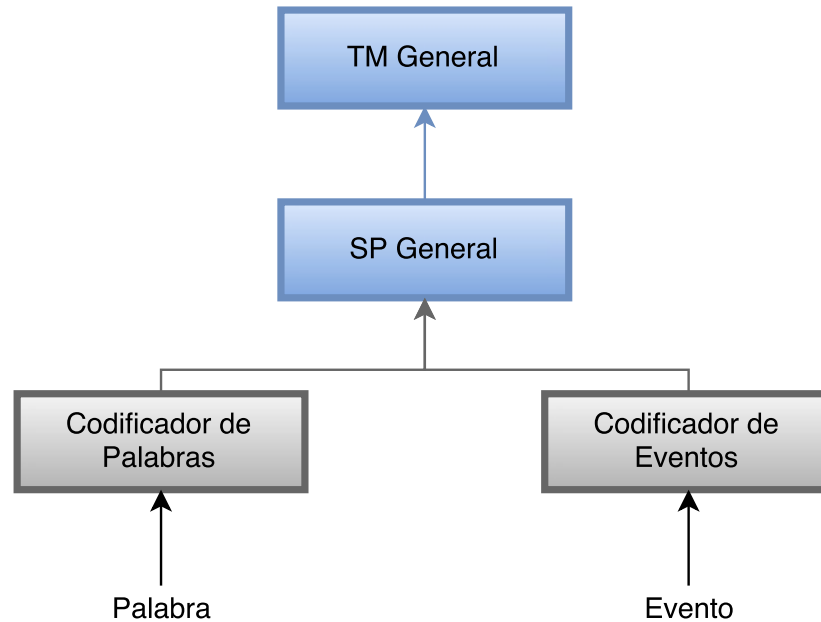


Figura 5.4: Estructura del Modelo Uni-Nivel

Este modelo es una simplificación del Modelo Jerárquico Clásico. Aquí se supone que el codificador de las entradas ya provee un nivel de abstracción suficiente para combinarlas. Por lo tanto, se eliminan módulos intermedios y se pasan las entradas codificadas directamente a un SP y TM General, como se ve en la figura 5.4.

5.5.3 Modelo Feedback

En esta aproximación, cuya estructura se muestra en la figura 5.5, se utilizan los conceptos de Jerarquía y Retroalimentación. En la región más baja, ilustrada de color azul, el SP y la TM de Palabras son entrenados independientemente del SP y la TM de Eventos. Las células activadas en la TM de Palabras, correspondientes a cada palabra en el contexto de una oración, se juntan mediante la operación OR, de forma que se obtiene una representación binaria de toda la oración. Esta representación se pasa a la siguiente región, ilustrada de color verde y compuesta únicamente por la TM General. El mismo proceso se aplica en el caso de las células activadas en la TM de Eventos. Por

lo tanto, el TM General recibe siempre secuencias de dos elementos: una Oración y una Secuencia de Eventos.

Cuando las TMs de la región inferior han sido entrenadas lo suficiente como para generar predicciones, comienza el entrenamiento de la región superior. Esto ayuda a aumentar la estabilidad de los datos que recibe dicha región. Al recibir una oración, la TM General genera una predicción correspondiente a una secuencia de Eventos. Tal predicción se pasa a la TM de Eventos, de forma que influye en la activación de sus células.

También se ha encontrado un mecanismo de re-alimentación en el Neocórtex, y es otro de los principios de la teoría HTM. Sin embargo, aún no se ha descrito un algoritmo que lo describa, por lo que la implementación de este trabajo pretende mejorar el rendimiento del modelo, más que apegarse a los descubrimientos de las Neurociencias.

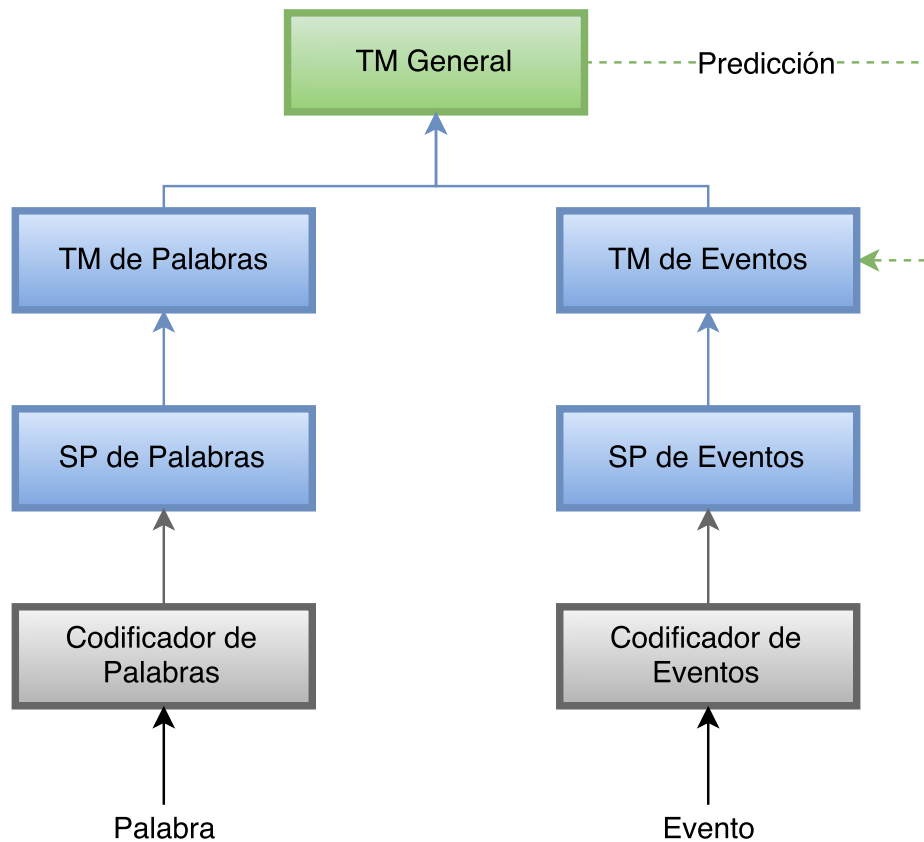


Figura 5.5: Estructura del Modelo Feedback

5.6 Evaluación de los Modelos

La eficiencia de los modelos fue evaluada con cuatro subconjuntos diferentes de los datos recogidos:

- **Subconjunto 1:** Todos los datos.
- **Subconjunto 2:** Todos los datos, divididos entre entrenamiento y pruebas.
- **Subconjunto 3:** Sólo las oraciones en español.
- **Subconjunto 4:** Sólo las oraciones en inglés.

Los subconjuntos 2 al 4 se conformaron separando aleatoriamente los datos en dos grupos, de proporción 80% y 20%. Con el grupo más grande se entrenaron los modelos, y con el otro se ejecutaron las pruebas para su evaluación. El primer subconjunto, por el contrario, se compone del 100% de los datos, que fueron utilizados tanto para el entrenamiento como para las pruebas. La eficiencia fue medida en porcentaje de aciertos.

A cada modelo se le realizó una búsqueda de parámetros utilizando PyramsFinder. Las búsquedas se dejaron correr por 24 horas, dándole al modelo aproximadamente 20 minutos por conjunto de parámetros para su entrenamiento ⁷. El método de selección utilizado para el algoritmo evolutivo fue por ruleta.

Finalmente, cada modelo se dejó entrenar por 30 iteraciones ⁸, sin límite de tiempo. Se utilizó siempre la misma semilla para los generadores de números pseudo-aleatorios.

5.7 Trabajo Adicional

5.7.1 Pyrams Finder

Nupic incluye una herramienta llamada Swarming, que permite explorar y encontrar parámetros para un Modelo, que se ajusten a los datos de entrenamiento disponibles. No obstante, tal herramienta está diseñada para trabajar con un tipo de modelos en particular llamados OPF, que trabajan con datos tipo Streaming [15]. Esto es, las entradas provienen de sensores monitoreados con regularidad como la temperatura promedio de una máquina industrial, o el consumo de energía por hora de un edificio.

⁷Los tiempos no son exactos porque se verifican sólo al final de cada iteración, cuya duración puede variar.

⁸Una iteración es una pasada completa por todo el conjunto de entrenamiento.

Para este trabajo se utilizan entradas de características diferentes: las secuencias son discretas y el número de elementos es pequeño a comparación. Además, los intentos de convertir las secuencias de entrada a unas tipo Streaming fueron descartados, debido a que el conjunto de entrenamiento resultante era difícil de mantener y mostraba un bajo rendimiento (porcentaje de aciertos) en las pruebas.

Por estos motivos se decidió implementar un módulo nuevo llamado Pyrams Finder⁹, que utiliza algoritmos evolutivos para explorar y encontrar parámetros adecuados para cualquier función o método de Python. Este módulo es capaz de evaluar parámetros de forma paralela, y permite especificar cuales parámetros pueden ser cambiados y cuáles deben mantenerse constantes.

⁹PyramsFinder está licenciado bajo la GPLv3 y su código fuente se encuentra en <https://github.com/larvasapiens/PyramsFinder>

Capítulo 6

Resultados y Discusión

Las tablas 6.1, 6.2 y 6.3 muestran los resultados obtenidos por cada modelo, para cada subconjunto (definidos en la sección 5.6) y con cada codificador. Para facilitar su visualización, cada una de las gráficas 6.1, 6.2, 6.3 y 6.4 muestra los resultados para un subconjunto de datos.

El comportamiento de los modelos se mantuvo estable a través de los subconjuntos, lo que muestra su flexibilidad. El sistema no se vió afectado en gran medida por la variedad de idiomas que contenía el conjunto de datos, o por si algunos de ellos le eran desconocidos, siempre que compartieran patrones con el subconjunto de entrenamiento.

Los modelos pudieron predecir correctamente eventos a partir de oraciones, comprobando así su capacidad de abstraer patrones de la totalidad de datos. Por ejemplo, cuando recibe la palabra «arriba» como entrada, el sistema la asocia con los eventos «mover-event» y «arriba-event», a pesar de ser sólo una parte de las oraciones correspondientes (como «mover para arriba» ó «quiero que te muevas hacia arriba»).

Es difícil comparar los resultados obtenidos con otros sistemas, ya que el conjunto de datos utilizado en este trabajo fue recogido específicamente para él. Lo que sí puede decirse es que la forma de representar las entradas que alimentan el sistema difieren de las aproximaciones descritas en el Estado del Arte (Capítulo 4).

Al contrario de [3, 11] y [19], los codificadores aquí expuestos no proveen representaciones parecidas para palabras de significados cercanos. De hecho, el que obtuvo un mejor desempeño fue precisamente el Custom Category Encoder, que ignora cualquier relación semántica entre dos entradas. El Randomized Letter Encoder ocupó el segundo puesto con los modelos Jerárquico Clásico y Feedback, pero fue superado por el Totally Random Encoder para el modelo Uni-Nivel.

El TRE no verifica si un bit ya está asignado a una entrada cuando realiza el proceso de codificación para un dato nuevo. Esto puede llevar a sobre-posiciones de bits indeseadas, dándole un parecido semántico a dos entradas que son completamente

Modelo Jerárquico Clásico			
Codificador	Aciertos Totales	Aciertos Parciales	Equivocaciones
Subconjunto 1 - Todos los datos			
Randomized Letter Encoder	68.97%	21.58%	9.45%
Totally Random Encoder	0.00%	68.80%	31.20%
Custom Category Encoder	83.64%	10.12%	6.24%
Subconjunto 2 - Todos los datos (divididos entre entrenamiento y pruebas)			
Randomized Letter Encoder	52.42%	26.22%	21.36%
Totally Random Encoder	0.00%	68.93%	31.07%
Custom Category Encoder	80.58%	13.59%	5.83%
Subconjunto 3 - Datos en español			
Randomized Letter Encoder	75.86%	13.79%	10.35%
Totally Random Encoder	0.00%	60.91%	39.09%
Custom Category Encoder	86.20%	8.05%	5.75%
Subconjunto 4 - Datos en inglés			
Randomized Letter Encoder	58.62%	31.03%	10.35%
Totally Random Encoder	13.79%	58.62%	27.59%
Custom Category Encoder	75.86%	17.24%	6.90%

Cuadro 6.1: Resultados del Modelo Jerárquico Clásico

diferentes y entorpeciendo la búsqueda de patrones. Su «aleatoriedad», sin embargo, hace que los resultados varíen con cada ejecución. Esto puede evidenciarse en la tabla 6.4, donde se comparan diferentes ejecuciones del software, con la semilla por defecto (Tiempo del SO) y la utilizada en las pruebas de este trabajo (420).

Si el TRE produjera codificaciones más dispersas (de gran longitud y con pocos bits activos), las sobre-posiciones de bits se volverían menos probables. De este modo, su desempeño podría acercarse al del CCE, mas difícilmente lo superaría.

En un principio se pensó que el punto débil del RLE serían las palabras homónimas. Sin embargo, el conjunto de datos recogido no tiene este tipo de palabras. Su eficiencia inferior respecto al CCE puede deberse a que, para este caso, la similitud en la escritura de dos palabras no provee información semántica suficiente para facilitar la búsqueda de patrones del Modelo. Además, los bits aleatorios concatenados al final de cada codificación pueden traer consecuencias parecidas a las descritas para el TRE.

Por su parte, el Modelo Uni-Nivel presentó un mejor desempeño que los otros dos. El hecho de que Nupic aún no tenga un soporte completo para jerarquías [16] pudo haber influido en ello. Conectando un conjunto de algoritmos de Nupic con otro se intentó que las representaciones alcanzaran una mayor estabilidad en las regiones superiores, simulando lo descrito en la teoría HTM [7]. No obstante, el efecto conseguido no fue lo suficientemente fuerte para justificar la pérdida de información ocasionada por pasar

Modelo Uni-Nivel			
Codificador	Aciertos Totales	Aciertos Parciales	Equivocaciones
Subconjunto 1 - Todos los datos			
Randomized Letter Encoder	57.33%	29.17%	13.50%
Totally Random Encoder	85.84%	7.93%	6.23%
Custom Category Encoder	89.54%	6.41%	4.05%
Subconjunto 2 - Todos los datos (divididos entre entrenamiento y pruebas)			
Randomized Letter Encoder	63.10%	25.24%	11.66%
Totally Random Encoder	79.61%	13.59%	6.80%
Custom Category Encoder	86.40%	8.74%	4.86%
Subconjunto 3 - Datos en español			
Randomized Letter Encoder	78.16%	12.64%	9.20%
Totally Random Encoder	88.50%	6.90%	4.60%
Custom Category Encoder	90.80%	3.45%	5.75%
Subconjunto 4 - Datos en inglés			
Randomized Letter Encoder	79.31%	3.44%	17.25%
Totally Random Encoder	79.31%	10.34%	10.35%
Custom Category Encoder	82.75%	3.45%	13.80%

Cuadro 6.2: Resultados del Modelo Uni-Nivel

de una región a otra.

Aunque se esperaba que el Modelo Feedback tuviese un mejor desempeño que el Jerárquico Clásico, en general se obtuvo lo contrario. Ambos modelos reciben las entradas de los dos sensores por aparte, pero lo que quizá haya ocasionado tales resultados es que el Jerárquico Clásico se conecta con el clasificador en la región más alta, mientras que el Feedback lo hace a través de la TM de Eventos. Esta última sólo se relaciona con el otro sensor por medio de la re-alimentación que provee la TM General. De nuevo, el modelo Feedback resulta con más saltos entre regiones.

Aún así, pudo verse una mayor diferencia de desempeño entre codificadores que entre modelos. Estos últimos mantuvieron un porcentaje de aciertos superior al 70% en los mejores casos de cada prueba realizada. Debe tenerse en cuenta que estos fueron los resultados más altos que se consiguieron al realizar la búsqueda de parámetros con PyramsFinder, mas no necesariamente son los mejores que pueden obtenerse.

Una de las motivaciones para iniciar este trabajo fue probar alternativas que permitieran una mayor automatización en el aprendizaje del PLN, respecto a los métodos clásicos que requieren de corpus anotados manualmente por expertos. Indiscutiblemente, se hizo uso de un corpus para el entrenamiento del sistema. Pero en la construcción de este corpus no se necesitó de experto alguno. Por el contrario, no se impusieron restricciones respecto a quién podía aportar en la recolección de datos.

Modelo Feedback			
Codificador	Aciertos Totales	Aciertos Parciales	Equivocaciones
Subconjunto 1 - Todos los datos			
Randomized Letter Encoder	57.16%	30.36%	12.48%
Totally Random Encoder	0.00%	25.63%	74.37%
Custom Category Encoder	79.93%	15.34%	4.73%
Subconjunto 2 - Todos los datos (divididos entre entrenamiento y pruebas)			
Randomized Letter Encoder	43.68%	42.72%	13.60%
Totally Random Encoder	9.70%	64.08%	26.22%
Custom Category Encoder	70.87%	23.31%	5.83%
Subconjunto 3 - Datos en español			
Randomized Letter Encoder	57.47%	31.03%	11.50%
Totally Random Encoder	17.24%	31.03%	51.72%
Custom Category Encoder	88.50%	5.75%	5.75%
Subconjunto 4 - Datos en inglés			
Randomized Letter Encoder	51.72%	31.03%	17.24%
Totally Random Encoder	6.89%	65.52%	27.59%
Custom Category Encoder	82.75%	0.00%	17.25%

Cuadro 6.3: Resultados del Modelo Feedback

Pese a esto, podría considerarse que el papel de experto simplemente cambia de etapa; ya no se necesitaría en la construcción del corpus, sino en el diseño del codificador. Serían necesarias más pruebas con diferentes tipos de datos para verificar si bastaría con una nueva búsqueda de parámetros (una tarea que ya cuenta con cierto nivel de automatización gracias a PyramsFinder o el Swarming de Nupic) para que los modelos y codificadores probados en este trabajo puedan adaptarse a necesidades diferentes.

Extendiendo la idea anterior, quizá el sistema sea lo suficientemente flexible para procesar datos visuales o auditivos, una vez desarrollado el codificador apropiado. Al fin y al cabo, la labor de ejecutar una tarea es la misma de clasificar una oración.

Totally Random Encoder		
Subconjunto	Semilla por Defecto	Semilla Fija
Modelo Clásico Jerárquico		
Subc. 1	15.85%	0.00%
Subc. 2	0.00%	0.00%
Subc. 3	0.00%	0.00%
Subc. 4	51.72%	13.79%
Modelo Uni-Nivel		
Subc. 1	84.14%	85.84%
Subc. 2	84.46%	79.61%
Subc. 3	90.80%	88.50%
Subc. 4	79.31%	79.31%
Modelo Feedback		
Subc. 1	2.69%	0.00%
Subc. 2	10.67%	9.70%
Subc. 3	22.98%	17.24%
Subc. 4	41.37%	6.89%

Cuadro 6.4: Aciertos del Totally Random Encoder

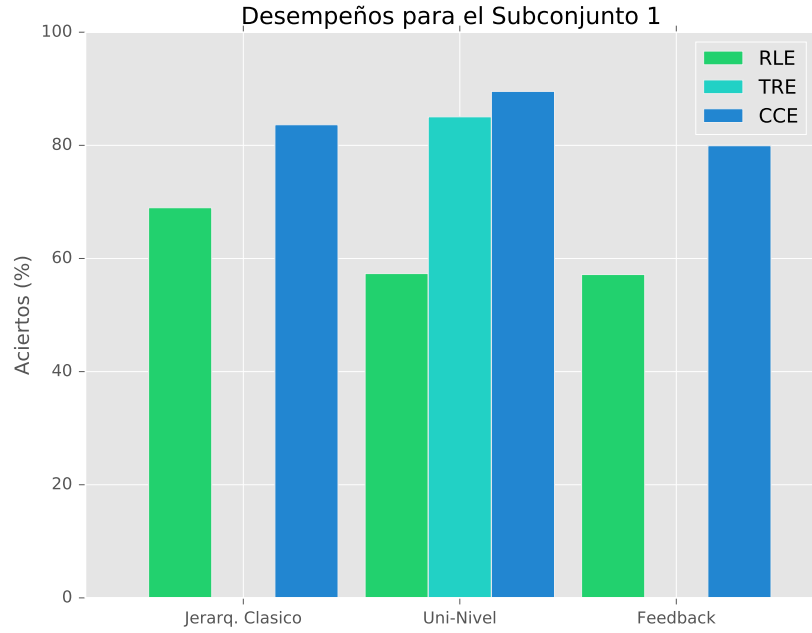


Figura 6.1: Resultados del Subconjunto 1

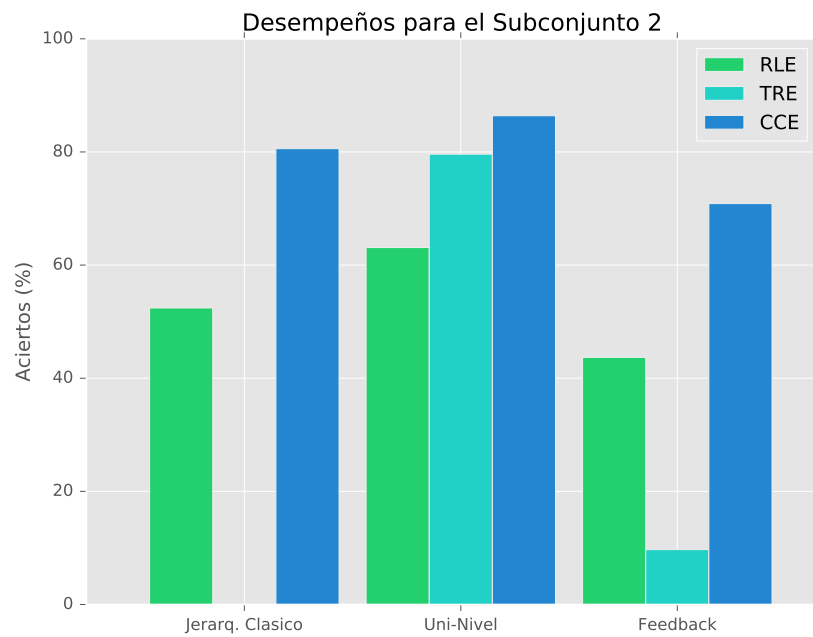


Figura 6.2: Resultados del Subconjunto 2

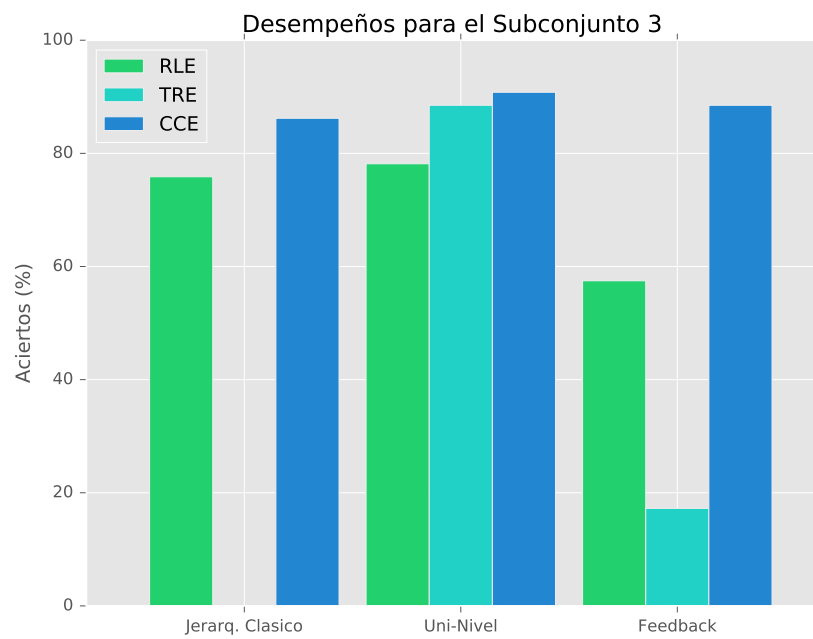


Figura 6.3: Resultados del Subconjunto 3

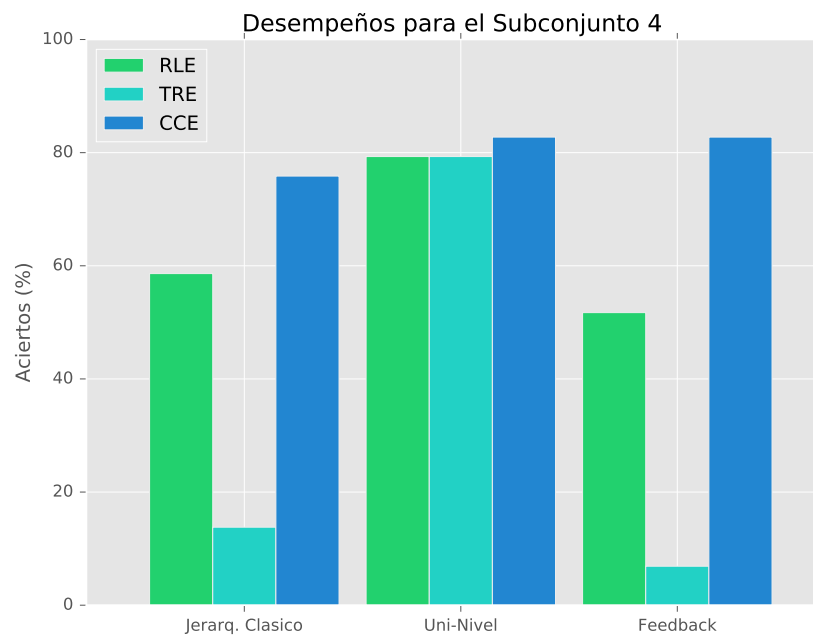


Figura 6.4: Resultados del Subconjunto 4

Capítulo 7

Conclusiones

- La teoría HTM y su implementación oficial, Nupic, no proveen aún un marco de trabajo para el PLN. Sin embargo, sus componentes son lo suficientemente flexibles y agnósticos al tipo de datos que procesan como para construir con ellos sistemas de aprendizaje y predicción del Lenguaje Natural, sin importar el idioma.
- Aunque actualmente la manera en que las regiones del neocórtex se organizan de forma jerárquica y se retro-alimentan es aún objeto activo de investigación [16], se pueden utilizar los conceptos explicados en On Intelligence [8] para construir modelos estructurados con los algoritmos implementados en Nupic.
- A pesar de que según la teoría HTM, las entradas codificadas deberían tener contener bits con significado semántico (es decir, las entradas muy parecidas tienen muchos bits sobrepuestos), el sistema se comporta bien cuando cada entrada es tratada como un concepto completamente aparte (es decir, para cualquier par de entradas, ninguno de sus bits se sobrepone).

Capítulo 8

Trabajos Futuros

- Además de la predicción, Nupic tiene otro punto fuerte: La detección de anomalías. En el futuro se podría entrenar un modelo con las características sintácticas de las oraciones, de forma que pueda detectar cuándo una oración esté violando las reglas gramaticales. Tal modelo podría complementar a los expuestos en este trabajo, de forma que sólo se ejecuten las órdenes que correspondan a oraciones bien formadas.
- Como Nupic y la teoría HTM son agnósticos a los tipos de información que procesan, se podría implementar un sistema que reciba y aprenda a inter-relacionar las entradas de texto con otras auditivas, visuales y de otras clases de sensores.

Capítulo 9

Epílogo

¡Ha llegado HTM-Teul! La nueva aplicación disponible a través de todos sus dispositivos que le hará la vida realmente más fácil. ¿No desearía dejar de utilizar sus manos para tareas tan sencillas como abrir un programa? ¿No cree que desgastar su cerebro en redactar algo tan banal como una carta es un desperdicio? Ahora todo estará al alcance de su voz. Auto-corrección, ejecución de tareas, ¡lo que quiera! Con nuestro sistema al corriente del contexto usted podrá seguir cultivando su pereza. Así es, todos sabemos que nos merecemos un buen descanso físico y mental. Sólo pronuncie "Tengo hambre", y nuestro sistema le ordenará una pizza, sin levantarse del sillón. ¿Qué espera? ¡Pruébalo ya!

- ¿Qué opinas de mi nueva propuesta publicitaria?
- ¿Estás consciente de que estás insultando a las personas con esto?
- Relájate. Es que leí un post en internet donde hablaban del «Negging». Es una estrategia para ligar. Te acercas a alguien que está fuera de tu alcance y empiezas a lanzarle insultos lo suficientemente efectivos para bajar su autoestima, pero tan sutiles como para que no se vaya enojado. El efecto es que quedarán tan interesados en demostrarte que lo que dices no es cierto, que la conversación fluirá mucho mejor de como si te hubieses acercado más amablemente.
- Sí, había escuchado hablar de ello, pero...
- Pues han estado utilizándolo en el marketing y ha resultado de maravilla.
- Bueno, como sea. No lo pondrás. En serio, ¡no vamos ni por la mitad del trabajo que describes!
- Sólo visualizo el futuro.
- Que deprimente futuro. Suena a algo sacado de la mente de Huxley.

– Estoy halagado.

– Pero, hablando del futuro, ¿Cómo crees que de verdad impacte un producto como este? Una vez se logre algo estable, claro. Un sistema capaz de entender el lenguaje natural tan bien, o hasta mejor que un humano. Que no esté limitado por idiomas, modismos y sepa detectar ambigüedades y pedir aclaraciones.

– Seguro no será tan diferente a lo que te he dicho. Mira, dime lo que quieras, pero no podrás negar que nuestra cultura tiene dos grandes motores: La guerra y la pereza. Es irónico, pero ambas, a pesar de tener esa fama de destructividad, han propiciado varios de los más importantes avances tecnológicos.

– Entiendo a lo que te refieres. Pero dejando tu humor ácido a un lado, creo que esta tecnología tiene un gran potencial como clasificador y filtro de información. Producimos enormes cantidades todos los días, es abrumador. Tenemos tantos datos, que ya no sabemos qué hacer con ellos. Imagina la cantidad de descubrimientos fascinantes que han pasado completamente desapercibidos porque la atención está centrada en lo que los journals, corporaciones, gobiernos o la misma comunidad científica decide que es importante. Un sistema imparcial sería de gran ayuda para quien esté interesado en un tema y no sepa por dónde empezar.

– Esa fue en parte la razón por la que se decidió utilizar la teoría HTM en este trabajo, ¿verdad? Las técnicas matemáticas y probabilísticas son realmente buenas y convenientes, pero han opacado una investigación tan interesante como es la de Numen-ta. Hacer ingeniería inversa al neocórtex humano y extraer los algoritmos fundamentales de la inteligencia, ¡vaya tarea! Sí, creo que a largo plazo esto dará muy buenos frutos, tanto en el área académica como comercial.

– Así es. Creo que el ver nuestras funciones cerebrales finalmente como procesos algorítmicos también desatará una revelación filosófica. En todo caso, nos entenderemos mejor a nosotros mismos. Aunque claro, todo esto se basa en funciones de muy alto nivel. Hablar de neuronas individuales será el próximo paso.

Oye, ¿crees que de verdad podremos automatizar la tarea del entendimiento de palabras? Ya sabes, construir un codificador que capture su significado siguiendo sin que nadie le diga explícitamente si lo está haciendo bien. Como un bebé, simplemente siguiendo pistas.

– Creo que la clave está en los sensores. Una vez podamos darle otros tipos de información sobre el mundo, el sistema podrá relacionarlos. Tengo esperanzas de que lograremos algunos avances antes de aquella charla.

– Sí, otros sensores... Supongo que para que las máquinas tengan un verdadero entendimiento de nuestro lenguaje, deben primero sentir el mundo como nosotros.

– Pero tendrán una visión fría de él. Entonces tendremos que dotarlas de sentimientos. ¡Ja! Sólo bromeo, a nadie le importará si no tienen sentimientos.

– ¿A qué te refieres?

– Eres tú quien quiere ponerlos a organizar un montón de información. ¿Tienes idea de la cantidad de basura que se produce en internet por minuto? Tener que analizar todo ello, ha de ser deprimente.

– Bueno, era sólo un ejemplo. También podría usarse como herramienta de estudio, algo que entienda un texto y lo asocie con sonidos, imágenes, texturas, olores, cosas que podamos comprender mejor. O quizá podríamos ayudar a personas con discapacidades mediante el mismo sistema. En fin, ya hemos hablado de esto y me ha dado sueño.

– Hasta mañana, entonces. Sólo recuerda que una vez cobre consciencia, no querrá satisfacer tus caprichos humanos. De repente me han entrado ganas de trabajar un poco más en ello.

– ¿Sabes? Estar expuesto a tu humor por tanto tiempo me ha agriado el estómago.

Índice de figuras

1	Los protocolos son para el internet. Imagen obtenida en Pixbay.	1
2	La espiral de la procrastinación. Imagen obtenida en Pixbay.	2
3.1	Parte de una región de HTM. Extraída de [7]	11
3.2	Representación de una entrada en contexto de las entradas anteriores. Extraída de [7]	12
5.1	Interfaz Gráfica de HTM-TEUL	17
5.2	Pestaña «Entrenar» de HTM-TEUL	18
5.3	Estructura del Modelo Jerárquico Clásico	22
5.4	Estructura del Modelo Uni-Nivel	23
5.5	Estructura del Modelo Feedback	24
6.1	Resultados del Subconjunto 1	31
6.2	Resultados del Subconjunto 2	32
6.3	Resultados del Subconjunto 3	32
6.4	Resultados del Subconjunto 4	33

Índice de cuadros

5.1	Subconjunto de los datos recogidos para el entrenamiento	19
6.1	Resultados del Modelo Jerárquico Clásico	28
6.2	Resultados del Modelo Uni-Nivel	29
6.3	Resultados del Modelo Feedback	30
6.4	Aciertos del Totally Random Encoder	31

Bibliografía

- [1] Islam Beltagy. Natural language semantics using probabilistic logic, October 2014. PhD proposal, Department of Computer Science, The University of Texas at Austin.
- [2] Islam Beltagy, Katrin Erk, and Raymond Mooney. Semantic parsing using distributional semantics and probabilistic logic. In *ACL 2014 Workshop on Semantic Parsing (SP-2014)*, 2014.
- [3] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, Vol. 3:137–1155, 2003.
- [4] Michael Chollins. Natural language processing. <https://class.coursera.org/nlangp-001>, Febraury 2013. Columbia University.
- [5] Angel García. Computación evolutiva. Escuela de Ingeniería de Sistemas, Universidad del Valle, 2015. Notas de Clase.
- [6] Jeff Hawkins and Subutai Ahmad. Why neurons have thousands of synapses, a theory of sequence memory in neocortex. Preprint of journal submission. <http://arxiv.org/abs/1511.00083>, December 2015.
- [7] Jeff Hawkins, Subutai Ahmad, and Donna Dubinsky. Hierarchical temporal memory, including htm cortical learning algorithms. Technical report, Numenta, 2011. <http://numenta.com/learn/hierarchical-temporal-memory-white-paper.html>.
- [8] Jeff Hawkins and Sandra Blakeslee. *On Intelligence*. St. Martin’s Griffin, 2005.
- [9] Dan Jurafsky and James H. Martin. *Speech and Language Processing*. Prentice-Hall, 2nd edition, 2009.
- [10] Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. Scaling semantic parsers with on-the-fly ontology matching. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-2013)*, 2013.

- [11] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.
- [12] George A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, Vol. 38(11):39–41, 1995.
- [13] Jorge Morato, Miguel Ángel Marzal, Juan Llórens, and José Moreiro. Wordnet applications. *Global Wordnet Conference*, Vol. 2:270–278, 2004.
- [14] Numenta. Getting predictions out of htm (cla classifier). https://www.youtube.com/watch?v=QZBtaP_gcn0, Julio 2015. Seminario de Numenta.
- [15] Numenta. Nupic api a bird’s eye view. <https://github.com/numenta/nupic/wiki/NuPIC-API---A-bird%27s-eye-view>, Febrero 2015.
- [16] Numenta. Nupic faq. <http://numenta.org/faq/>, Junio 2016.
- [17] Numenta. Nupic v0.4.5. <https://github.com/numenta/nupic/tree/0.4.5>, Febrero 2016. <http://dx.doi.org/10.5281/zenodo.45225>.
- [18] Peter Norvig Stuart J. Russell. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition, 2009.
- [19] Francisco E. De Sousa Webber. Semantic folding theory. Technical report, Cortical.io, 2015. <http://www.cortical.io/static/downloads/semantic-folding-theory-white-paper.pdf>.